APPLICATION

FOR

UNITED STATES LETTERS PATENT

TITLE: UNDOING USER ACTIONS IN A CLIENT PROGRAM

APPLICANT: MALTE WEDEL AND ANDREAS ROESSLER

CERTIFICATE OF MAILING BY EXPRESS MAIL

Express Mail Label No. EV 321 388 354 US

September 30, 2003
Date of Deposit

UNDOING USER ACTIONS IN A CLIENT PROGRAM

BACKGROUND

The present invention relates to data processing by digital computer, and more particularly to undoing user actions in a client program.

Client-server applications typically include an application component running on a server, and a client program running on a client computer. The client program renders a user interface through which a user can interact with the application. The user interface can contain one or more controls. A control is a user interface element through which a user interacts with, provides input to, or controls an application. Examples of controls are text fields, radio buttons, tables, trays, and drop-down menus.

Once a user provides input to a control, the user may at a later time wish to undo the input. However, conventional undo techniques for client-server applications are limited in functionality. For example, the undo functionality may only be available for text fields and only available while the input focus remains on the text field.

SUMMARY OF THE INVENTION

The present invention provides methods and apparatus, including computer program products, implementing techniques for undoing user actions in a client program.

In general, in one aspect, the techniques include displaying a user interface in a client program, the user interface having a plurality of controls, the plurality of controls including multiple types of controls, each control having a state; for each control in the plurality of controls, storing the state of the control as a first state for the control; receiving user input comprising a change to the state of a control in the plurality of controls; updating the state of the control based on the user input; storing the updated state of the control as a second state for the control; receiving user input comprising a request to undo the change; and restoring the state of the control to reflect the first state for the control.

Advantageous implementations of the invention include one or more of the following features. The multiple types of controls include one or more of a text field control type, a radio button control type, a table control type, a tray control type, and a menu control type.

The user input comprising the request to undo the change is received while focus is not on the control.

The state of the control includes a data state and a view state. The operations further include determining whether the change affects the data state or the view state of the control; and restoring the state of the control only if the change affects the data state of the control. The operations further include receiving user input comprising a request to redo the change to the control; and restoring the state of the control to reflect the second state for the control.

Restoring the state of the control includes restoring the state of another control that shares data with the control. Restoring the state of the control includes restoring the state of another control that shares data with the control. Restoring the state of the control occurs prior to transmitting the state of the control to a server.

In general, in another aspect, the techniques include generating at least one data structure that stores application data, and associations between the application data and one or more application controls that are rendered based on the stored application data; detecting that the at least one data structure has changed from a prior state to a new state; recording the prior state of the at least one data structure; receiving user input requesting that an undo operation be performed; and performing the undo operation by restoring the at least one data structure to the prior state.

Advantageous implementations of the invention include one or more of the following features. The at least one data structure is at least one data tree. The at least one data structure is stored on a client device. The application controls include multiple types of controls. The associations between the application data and the application controls are defined by metadata for the application.

The invention can be implemented to realize one or more of the following advantages. The undo mechanism can run in a web browser. The undo mechanism can operate on different types of controls. The undo mechanism can be invoked to undo user actions or to restore user input to a control even after the input focus has shifted away from the control. The scope of the undo mechanism can be changed so as to undo only certain types of user modifications (e.g., modifications that affect data, or modifications that affect formatting).

The undo mechanism can be used to undo user actions at different levels of granularity. When the undo mechanism is invoked to undo a change (e.g., to undo user action or restore user input) for a particular control, and that control shares data with another control, the change is undone for both controls, thereby maintaining data consistency.

The undo mechanism does not need to be coded independently for each control in the client program. Because the undo mechanism is independent from the controls, the undo mechanism can automatically extend to new controls that are added to the client program. One implementation of the invention provides all of the above advantages.

The details of one or more implementations of the invention are set forth in the accompanying drawings and the description below. Further features, aspects, and advantages of the invention will become apparent from the description, the drawings, and the claims.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1 is a block diagram of a system in accordance with the invention.

FIG. 2 is a flow diagram of a method in accordance with the invention.

FIG. 3 is a block diagram of one implementation of the system.

FIG. 4 is a block diagram of one implementation of the system.

Like reference numbers and designations in the various drawings indicate like elements.

DETAILED DESCRIPTION

As illustrated in FIG. 1, a system 100 in accordance with the invention includes a client program 110 running on a client computer. The client program 110 renders a user interface 120 through which a user can interact with an application that is running on a server computer. The user interface 120 contains one or more controls 130. The controls 130 include multiple types of controls including text fields, radio buttons, table controls, trays and menus.

Each control 130 has a state that can include a view state and a data state. The visual properties of the control define the view state of the control. Examples of visual properties include the visibility of a tray control (e.g., expanded or collapsed), or the scroll position of a

table control. Data associated with the control defines the data state of the control. The data can include different types of data including strings, booleans, or objects (e.g., a Java.util.date object).

The system also includes an undo mechanism 140 that can be invoked by a user to undo or redo prior user actions taken with respect to the controls before the user actions have been transmitted to the server. For example, the undo mechanism 140 can be invoked to undo a delete operation performed by the user with respect to a control, thereby restoring prior input provided by the user to that control. The undo mechanism 140 can also provide a redo operation that negates the effect of the undo operation. The undo mechanism 140 can be integrated as part of the client program 110 or alternatively can be separate from the client program 110.

The undo mechanism 140 is operable to record the initial state of the controls and to detect subsequent changes in the state of the controls. The state of a control can change as a result of user interaction with the control, for example, user entry of data into a text field, user selection of a radio button, or user manipulation of a scroll bar. The undo mechanism 140 is operable to store the initial state and the subsequent states of each control in one or more lists 150. A list 150 can be implemented using an ordered data structure such as a stack or a queue, with the most current state being at the top of the stack or the front of the queue. In one implementation, a pointer is used to identify the stack entry that corresponds to the current data state. Initially, the pointer points to the top of the stack. As each undo operation is performed, the pointer position moves down the stack. As each redo operation is performed, the pointer position moves up the stack. When new user input is received, any entries between the current pointer position and the top of the stack are removed, a new entry corresponding to the new user input is added to the new top of the stack, and the pointer position is set to the new entry. Once the state of the controls has been transmitted to the server, the stack is cleared.

In operation, as illustrated by the method 200 of FIG. 2, the system 100 stores the initial state of the controls in the list 150 (step 210). The system receives user input that changes the state of one or more of the controls 130 (step 220). The system stores the state

change in the list 150 (step 230). Subsequently, the system receives user input requesting to undo the change (step 240). The input focus need not be on the control to be changed when the user makes such a request. For example, the user can enter data into a first text field, press the tab key to establish focus on a second text field, and then invoke undo to undo the change to the first text field.

In response to the undo request, the system invokes the undo mechanism 140 to undo the change (step 250). The undo mechanism uses the undo list 150 to restore one or more of the controls to the state immediately prior to the current state. In the pointer implementation described above, restoring one or more of the controls to the state immediately prior to the current state involves repositioning the pointer so that it points to the stack entry immediately below the stack entry previously pointed to.

The mechanism can also redo a state change in response to user input requesting such an action (step 260). In the pointer implementation described above, the undo mechanism performs a redo operation by repositioning the pointer so that it points to the stack entry immediately above the stack entry previously pointed to.

Scope and granularity settings

In some implementations, the undo mechanism 140 can be configured according to a variety of settings, including scope settings and granularity settings. Scope settings can be used to indicate the types of changes that can be undone. For example, the undo mechanism 140 can be configured to undo all state changes (e.g., both view state changes and data state changes), or only certain kinds of state changes (e.g., data state changes only). In such an implementation, the undo mechanism 140 can distinguish between data state changes and view state changes, and if only the data state changes are to be undone, then the undo mechanism does not need to store the view state changes in the list 150.

Granularity settings can be used to specify the level at which user actions to be undone and/or redone. For example, the granularity level can specify that user actions are to be undone at a character, string, or field level. A character-level undo undoes the last character that was entered. For example, if the last user input were a character string "Hello", a character-level undo would only undo the "o" character, but not the entire string. A

string-level undo undoes not only the last character, but also the entire string that includes the last character. Thus, in the example above, the entire string "Hello" would be undone. A field-level undo treats an entire field as an atomic unit. For example, a field can include one or more character strings as well as other input such as TAB or Enter keystrokes. A field-level undo would undo all the input in the field, not just a single string or a single character.

In one implementation, the field-based undo is implemented by monitoring a data tree, as described below. For other granularities, the undo mechanism can be implemented with a different monitoring process. For example, a character-based undo mechanism can be implemented by monitoring the user input events detected by the system.

Data Trees

In one implementation of the system 100, the undo mechanism detects changes to the controls by monitoring one or more data trees 310. As shown in FIG. 3, the one or more data trees 310 store the data associated with each control 130. Each control 130 includes a reference to the node in the data tree 310 where the control's data is stored.

The data stored in the data tree includes state information that can include both view state and data state information for the controls. Upon detecting a change in the view state or data state of a control, the corresponding node in the data tree is updated to reflect the change. The methods used to update the view state (view state update methods) can differ from the methods used to update the data state (data state update methods).

A monitoring process 320 monitors the data tree for changes and records the changes in the list 150. The monitoring process can record all changes; alternatively, it can distinguish between view state and data state changes and only record the data state changes. In an implementation in which different methods are used to update view states and data states, the monitoring process can distinguish between view state and data state changes by determining whether view state update methods or data state update methods were invoked to update the data tree.

The undo mechanism uses the list 150 generated by the monitoring process to keep track of the current and past states of the data tree. When a user requests an undo operation be performed, the undo mechanism can use the list 150 to restore the data tree to a prior state.

Framework

In one implementation of the system 100, as shown in FIG. 4, the client program 110 is a Web browser 420, the user interface 120 is a Web page 430, and the undo mechanism 140 is provided by a software framework 410 running in the Web browser 420.

The Web browser 420 communicates with the server using HTTP (Hypertext Transfer Protocol). HTTP is a stateless protocol, meaning that each time the Web browser 420 requests a Web page 430, the server will respond to the request independently of any previous requests by the client device. The server response generally includes code, for example, HTML (Hypertext Markup Language) code, that specifies how to render the Web page 430. Rendering the Web page 430 can involve generating a document object model (DOM) representation of the Web page 430.

The server response can also include code for establishing the framework 410 in the Web browser 420. The framework code can include client-side scripting code such as, for example, JavaScript or VBScript code. The framework code can be embedded in the code for the Web page 540 or stored as a separate file that is referenced by the code for the Web page 430. The framework code can be generated based on metadata for the application that is running on the server.

Once established, the framework 410 includes one or more data structures (e.g., data trees) that store the state of each control in the Web page 430. The framework 410 monitors the changes to the state of each control by monitoring the one or more data structures. The framework 410 records the state changes in a list which can then be used by the undo mechanism 140 to undo the changes as described above. The undo mechanism in such an implementation is independent of the controls, which means that the undo mechanism can automatically provide undo functionality for new controls (or new types of controls) added to the client program without requiring code for an undo function for each new control (or each new type of control).

In one implementation, each data element appears only once in the data structures, even if the data element is associated with more than one control. For example, the data element that stores zip code information can be associated with a first text field that displays the zip code and a second text field that displays the city name. When the undo operation is invoked to restore the value of the zip code field, the city name field will be restored as well.

The invention can be implemented in digital electronic circuitry, or in computer hardware, firmware, software, or in combinations of them. The invention can be implemented as a computer program product, i.e., a computer program tangibly embodied in an information carrier, e.g., in a machine-readable storage device or in a propagated signal, for execution by, or to control the operation of, data processing apparatus, e.g., a programmable processor, a computer, or multiple computers. A computer program can be written in any form of programming language, including compiled or interpreted languages, and it can be deployed in any form, including as a stand-alone program or as a module, component, subroutine, or other unit suitable for use in a computing environment. A computer program can be deployed to be executed on one computer or on multiple computers at one site or distributed across multiple sites and interconnected by a communication network.

Method steps of the invention can be performed by one or more programmable processors executing a computer program to perform functions of the invention by operating on input data and generating output. Method steps can also be performed by, and apparatus of the invention can be implemented as, special purpose logic circuitry, e.g., an FPGA (field programmable gate array) or an ASIC (application-specific integrated circuit).

Processors suitable for the execution of a computer program include, by way of example, both general and special purpose microprocessors, and any one or more processors of any kind of digital computer. Generally, a processor will receive instructions and data from a read-only memory or a random access memory or both. The essential elements of a computer are a processor for executing instructions and one or more memory devices for storing instructions and data. Generally, a computer will also include, or be operatively coupled to receive data from or transfer data to, or both, one or more mass storage devices for

storing data, e.g., magnetic, magneto-optical disks, or optical disks. Information carriers suitable for embodying computer program instructions and data include all forms of non-volatile memory, including by way of example semiconductor memory devices, e.g., EPROM, EEPROM, and flash memory devices; magnetic disks, e.g., internal hard disks or removable disks; magneto-optical disks; and CD-ROM and DVD-ROM disks. The processor and the memory can be supplemented by, or incorporated in special purpose logic circuitry.

To provide for interaction with a user, the invention can be implemented on a computer having a display device, e.g., a CRT (cathode ray tube) or LCD (liquid crystal display) monitor, for displaying information to the user and a keyboard and a pointing device, e.g., a mouse or a trackball, by which the user can provide input to the computer. Other kinds of devices can be used to provide for interaction with a user as well; for example, feedback provided to the user can be any form of sensory feedback, e.g., visual feedback, auditory feedback, or tactile feedback; and input from the user can be received in any form, including acoustic, speech, or tactile input.

The invention can be implemented in a computing system that includes a back-end component, e.g., as a data server, or that includes a middleware component, e.g., an application server, or that includes a front-end component, e.g., a client computer having a graphical user interface or a Web browser through which a user can interact with an implementation of the invention, or any combination of such back-end, middleware, or front-end components. The components of the system can be interconnected by any form or medium of digital data communication, e.g., a communication network. Examples of communication networks include a local area network ("LAN") and a wide area network ("WAN"), e.g., the Internet.

The computing system can include clients and servers. A client and server are generally remote from each other and typically interact through a communication network. The relationship of client and server arises by virtue of computer programs running on the respective computers and having a client-server relationship to each other.

The invention has been described in terms of particular embodiments. Other embodiments are within the scope of the following claims. For example, the steps of the invention can be performed in a different order and still achieve desirable results.

What is claimed is: